**Name:**

**Advanced Programming in Java**

**Lab Exercise 9/30/2019**

**Working with GUI Components In Java**

In Java there are two vehicles for adding GUI interface components; Abstract Windowing Toolkit (AWT) and Swing. In order to use these, you must have the following:

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

The first step in creating an application that uses these GUI components is to create a Window. The Window that we are going to be using is called a JFrame (components that start with J can be considered to be Swing components).

Once you have a Window we will then create a Container to hold our GUI components. The Container class is part of the AWT.

After we have defined a Container, we will need to define a FlowLayout for that Container.

Now that we have a FlowLayout defined, we may add whichever components we require.

Here is a sample program demonstrating this (create to files TestWindow.java and Test.java to hold the TestWindow and Test classes):

//TestWindow.java

//Class definition of TestWindow class

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

class TestWindow extends JFrame // inherits from JFrame

{

public TestWindow()

{

// Use these four methods in every application constructor

super("Test Application");

setSize(300, 100);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setVisible(true);

// Create the container

Container contentArea = getContentPane();

contentArea.setBackground(Color.white);

// Define flow layout

FlowLayout flowManager = new FlowLayout();

contentArea.setLayout(flowManager);

// add GUI components to the Container

JButton testButton = new JButton("Test");

contentArea.add(testButton);

// add other controls as necessary

// Apply components to the Container

setContentPane(contentArea);

} // end of constructor

}

// Test.java

// Driver program to test the Test class

public class Test

{

public static void main(String[] args)

{

TestWindow whatever = new TestWindow();

}

}

In this example, we have used the default FlowLayout. It should be noted that there are other options such as the GridBagLayout.

Once we have built our interface, we will want to make out application interactive and respond to events. The applicaton must be made to “listen” for events utilizing one of the Java EventListener interfaces. Interfaces are invoked utilizing the keyword *implements*. For example:

class TestWindow extends JFrame implements ActionListener

Different event listeners are utilized to detect different types of events. The following table summarizes”

|  |  |
| --- | --- |
| **EventListener** | **Event** |
| ActionListener | Button clicks |
| ItemListener | Checkbox item selections  Radio button item selections  Combo box item selections |
| KeyListener | Keyboard input |
| MouseListener  MouseMotionListener | Mouse actions  Mouse movements |
| ChangeListener | Slider changes |

Each EventListener must be added to the item it is listening on. For example:

JButton testButton = new JButton("Test");

testButton.addActionListener(this);

contentArea.add(testButton);

The following table summarizes the method to add an EventListener:

|  |  |
| --- | --- |
| **EventListener** | **Method to add** |
| ActionListener | name.addActionListener(this) |
| ItemListener | name.addItemListener(this) |
| KeyListener | name.addKeyListener(this) |
| MouseListener | name.addMouseListener(this) |
| MouseMotionListener | name.addMouseMotionListener(this) |
| ChangeListener | name.addChangeListener(this) |

Once the EventListener is added to the component, the event handler must be written. For example:

public void actionPerformed(ActionEvent event)

{

// write event handler code here

}

The following table summarizes the event handlers that must be implemented with each EventListener

|  |  |
| --- | --- |
| **EventListener** | **Event Handler** |
| ActionListener | public void actionPerformed(ActionEvent event) |
| ItemListener | public void itemStateChanged(ItemEvent event) |
| KeyListener | public void keyTyped(KeyEvent event)  public void keyPressed(KeyEvent event)  public void keyReleased(KeyEvent event) |
| MouseListener | public void mousePressed(MouseEvent event)  public void mouseReleased(MouseEvent event)  public void mouseClicked(MouseEvent event)  public void mouseEntered(MouseEvent event)  public void mouseExited(MouseEvent event) |
| MouseMotionListener | public void mouseMoved(MouseEvent event)  public void mouseDragged(MouseEvent event) |
| ChangeListener | public void stateChanged(ChangeEvent event) |

Important Note: If you implement an EventListener interface, you must implement all of the event handlers even if they do not do anything. For example, if you implement a MouseListener you must implement all five of it’s event handlers.

Here is an example of a button and a textbox.

// Actions.java

// Actions class definition

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

class Actions extends JFrame implements ActionListener

{

JTextArea textarea = new JTextArea(2,25);

JButton button1 = new JButton("Button #1");

JButton button2 = new JButton("Button #2");

public Actions()

{

super("Action Events");

setSize(300, 100);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setVisible(true);

Container content = getContentPane();

FlowLayout layout = new FlowLayout();

content.setLayout(layout);

button1.addActionListener(this);

button2.addActionListener(this);

content.add(button1);

content.add(button2);

content.add(textarea);

setContentPane(content);

}

public void actionPerformed(ActionEvent event)

{

String str="From: "+event.getActionCommand();

if(event.getSource()==button1) str +=" - No.1";

if(event.getSource()==button2) str +=" - No.2";

textarea.setText(str);

}

}

// TestActions.java

// Driver program to test the Actions class

class TestActions

{

public static void main(String[] args)

{

Actions eg = new Actions();

}

}

**Today’s Project**

On the server you will find a completed project JCalculator. It is a Java Applet. You task is to port this to a Java Application that runs in Windows as opposed to running on a Web Browser.